Practice Questions on Sorting and 1-dimensional Dynamic Programming

These questions (and the ones listed under readings, and the ones in the previous two sets of practice questions) are good preparation for Test #1. They are not to be handed in. Some hints follow the questions, but don't look at them unless you are stuck after working on the problem for a while.

1. A point in the plane can be represented using Cartesian coordinates (x, y). The distance of (x, y) from the origin is $\sqrt{x^2 + y^2}$. Suppose we wish to sort a set of n points $(x_1, y_1), \ldots, (x_n, y_n)$ according to their distance from the origin.

(a) Suppose the two coordinates of each of the n points are integers between $-n$ and n. Describe how to sort the points according to their distance from the origin in $O(n)$ time in the worst case. Briefly explain why your answer is correct.

(b) Now consider n points whose coordinates are floating point numbers. Assume the points are drawn uniformly, randomly and independently from the disc given by $x^2 + y^2 < 1$. How can you adapt bucket sort to sort the points according to their distance from the origin in $O(n)$ expected time? Be explicit about how you calculate the index of the bucket that each point belongs to.

2. Recall the canoe rental problem from class. There are n canoe rental agencies numbered $1..n$ along a river. The price of renting a canoe at agency i and returning it at agency $j > i$ is $p_{i,j}$. Let $P[i]$ is the minimum possible total price for travelling from i to n. A dynamic programming solution to compute $C[i]$ is to use the following recurrence:

$$\begin{aligned} P[n] &= 0 \\ P[i] &= \min_{i<j\leq n} (p_{i,j} + P[j]) \end{aligned}$$

Consider the following different variation of the problem. The maximum capacity canoe that can be rented at agency i and returned at agency $j > i$ can carry $c_{i,j}$ kg. Joan wants to take a trip from agency 1 to agency n with the maximum possible weight allowance.

(a) Joan comes up with the following solution. At each agency along the way, always choose the next destination according to the one that allows the biggest possible capacity canoe that agency i rents to be taken. (In other words, if she is at agency i, she chooses the next destination j to be the one that maximizes $c_{i,j}$.) Show that this solution does *not* yield an optimal solution.

(b) Use dynamic programming to solve the problem. How can you print the optimal itinerary for Joan?

3. Given a sequence of integers $a_1, a_2, \ldots, a_n$ you want to find a longest increasing subsequence. For example, a longest strictly increasing subsequence of $\langle 4, \mathbf{1}, 7, \mathbf{2}, 9, 8, \mathbf{3}, \mathbf{8}\rangle$ is the subsequence $\langle 1, 2, 3, 8\rangle$ shown in bold. Use dynamic programming to first find the length of the subsequence and then print the subsequence.

Hints

1.

(a) Sorting by the square of the distance is the same as sorting by distance. How many possible values can there be for the square of the distance?

(b) One way to solve this is to ensure that approximately the same number of elements fall into each bucket. Where should the dividing lines between buckets be?

2.

(a) It should be possible to find a set of capacities where Joan's solution yields a suboptimal solution by playing around with a few small examples.

(b) Modify the dynamic programming solution in class. Think about what needs to be changed. Recall that the steps of coming up with a dynamic programming solution are:

- think of the solution as a sequence of choices to be made, and focus on the first choice to determine which subproblems need to be solved
- define what each array entry is supposed to represent
- come up with a recurrence to compute entries of the array
- decide what order to fill in entries of the array so that whenever you need an array entry, it is already computed
- figure out which entry of the array provides the solution to the original subproblem

3. The subproblems you should consider are finding the length of the longest increasing subsequence that starts with item a_i.